Université Nice Sophia Antipolis

i3S sophia antipolis

CNRS

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

INRIA

centre de recherche SOPHIA ANTIPOLIS - MÉDITERRANÉE

Clock Constraint Specification Language

semantics & graphical representation

Aoste Project

UML & FM - Kita Kyushu -
September 27th, 2008

Motivation (1/2)

- **Software** systems need
  - tools to **design**, **manipulate** and **store** *engineering information* (functional and extra-functional specifications, resources, binaries…)
  - ➢ Various diagrams (structural, behavioral, deployment, …)

UML provides a framework to merge all kinds of software models

- **Real-Time and Embedded** (RTE) systems need
  - tools to **describe**, **manipulate** and **analyze** *interactions*, *communications*, *synchronizations* between processes.
  - ➢ Process algebras, models of computation and communication

UML should also provide a framework to merge concurrency models

Motivation (2/2)

- In the real world, SW and RTE designers
  - Use UML to draw graphs, vertices and edges, with fancy adornments
  - Perform model transformations to their proprietary language that makes its own assumptions and give its own semantics
  - ➢ Models are not merged but only stored in the same bundle

- MARTE defines a common ground (and semantics?) for building RTE models with UML
  - The MARTE Time model relies on CCSL to define interactions among *clocks* (processes, actors, …)
  - MARTE should be extended for domain-specific purposes

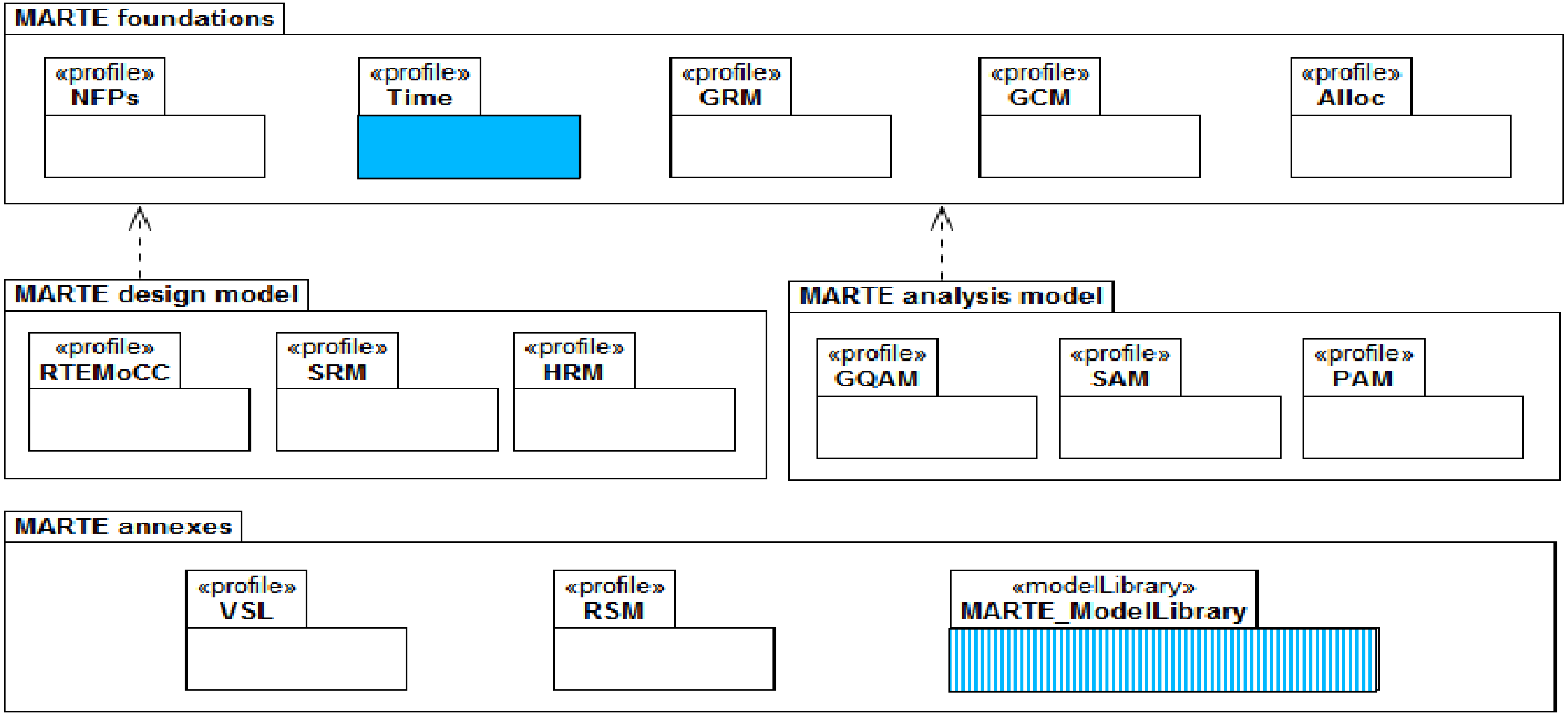
Where to put the semantics itself ? In the OMG specification ?

Outline

- Overview of representative CCSL operators
  - Synchronous, Asynchronous, Mixed
  - Quantitative
- Discussion on possible graphical representations for CCSL
  - UML constraints
  - UML Profile for CCSL
  - parametric-like clock diagrams (Inspired from SysML)
- Questions

About MARTE

- OMG UML2 Profile for **M**odeling and **A**nalysis of **R**eal-**T**ime and **E**mbedded systems
  - OMG Adopted Specification (ptc/07-08-04) => FTF



About MARTE

- OMG UML2 Profile for **M**odeling and **A**nalysis of **R**eal-**T**ime and **E**mbedded systems
  - OMG Adopted Specification (ptc/07-08-04) => FTF

- Time Model
  - Define a Timed Causality Model for UML
  - Broad enough to give the semantics of various domain-specific formalisms (AADL, IP-Xact, East-ADL2, …)

Time model - Clocks

- Any event (start/end of actions; send/receive of messages; transition being fired; …) is a **Clock**
  - When the *distance* between two successive occurrences of the event is meaningful (like in Physical time) => Chronometric clocks
  - Otherwise => **Logical clocks**

- More formally, a clock is a five-tuple ⟨A, e , < , λ, u⟩
  - A is a set of instants (possibly infinite);
  - e is a strict quasi-order relation on A;
  - < is as set of labels;
  - λ : A →< is a labeling function ;
  - u is the unit.

- Clocks can be
  - *discrete* (A is a discrete set) - idx : A →¢*, idx is order-preserving
  - or *dense*.

Today, focus on discrete logical clocks

Time model – Time structure

- Several interdependent clocks are gathered within a **time structure**

- A *time structure* is a pair $\langle ;, y \rangle$
  - $;$ is a finite set of clocks;
  - y is a partial order relation on $\bigcup_{c\in ;} A_c$

- From y we derive four *instant relations*:
  - *Coincidence*: $\equiv \triangleq y \cap z$
  - *Strict precedence*: $e \triangleq y \setminus \equiv$
  - *Independence*: $\parallel \triangleq \overline{y \cup z}$
  - *Exclusion*: $\# \triangleq e \cup [$

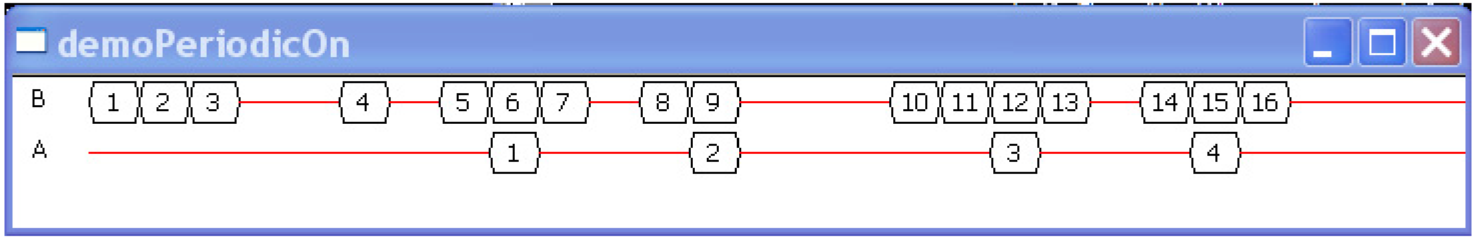
Time model – Clock relations

- *Clock relations* define (infinitely) many instant relations
- Four categories of clock relations
  - **Coincidence-based** (synchronous)
    - isSubClock, discretizedBy, **isPeriodicOn**, filteredBy …
  - **Precedence-based** (asynchronous)
    - isFasterThan (precedes), **alternatesWith** …
  - **Mixed** (asynchronous => synchronous)
    - **sampledOn**, delayedFor, timer, inf, sup …
  - **Quantitative** (related to chronometric clocks)
    - hasStability, hasOffset, hasJitter, hasDrift …
- Clock Constraint Specification Language = concrete syntax
  - Non-normative annex of MARTE

Clock constraint - Coincidence

- Logical periodicity
  - A **isPeriodicOn** B period=P offset=δ

    $(\forall k \in \text{¢}^{*})\ (A[k] \equiv B[(k-1).P + \delta + 1])$



Period=3
Offset=5

- Chronometric periodicity
  - A ≡ IdealClk **discretizedBy** P

    $(\forall k \in \text{¢}^{*})\ (A[k+1] - A[k] = P)$

Clock constraint - Precedence

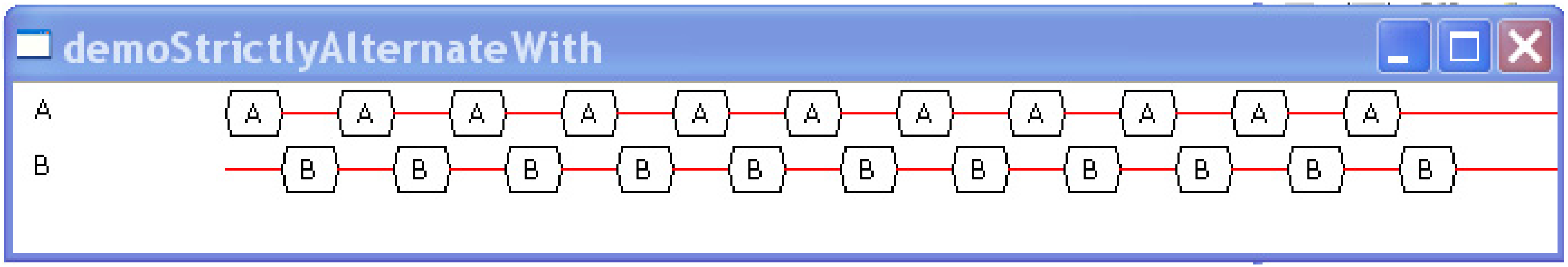
- A **isFasterThan** B (A e B), B **isSlowerThan** A

  (∀k∈¢*) (A[k] e B[k]) (strict form)

- A **strictly alternatesWith** (A 0 B)

  (∀k∈¢*) (A[k] e B[k] e A[k+1])

  ⇔ A e B e (A<<1)

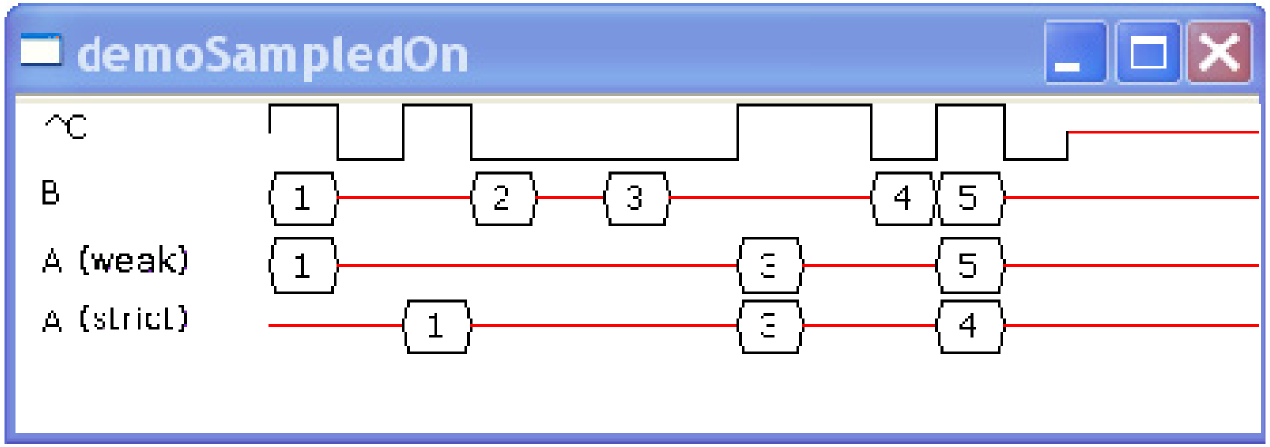


- asynchronous communications or causal relations

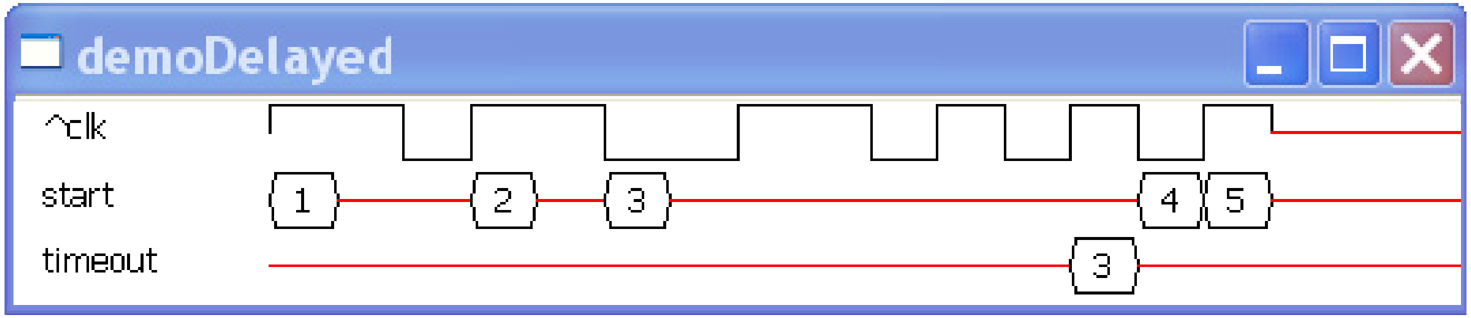
Clock constraint - Mixed

- A = B **sampledOn** C (A = B⚡C)

(∀a∈¢*) (∃b,c∈¢*) (A[a] ≡ C[c]) ∧ (C[c-1]ℯ B[b] y C[c])



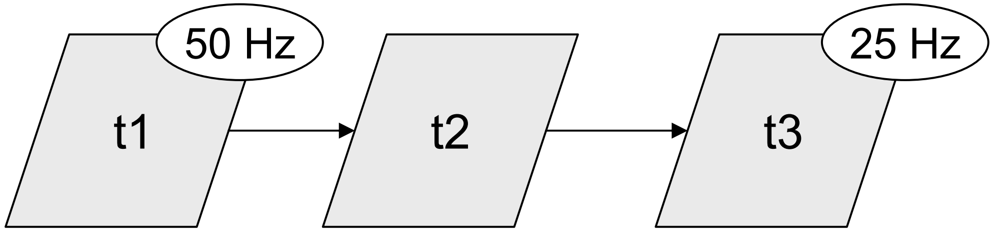
- A = B **delayedFor** δ **on** C

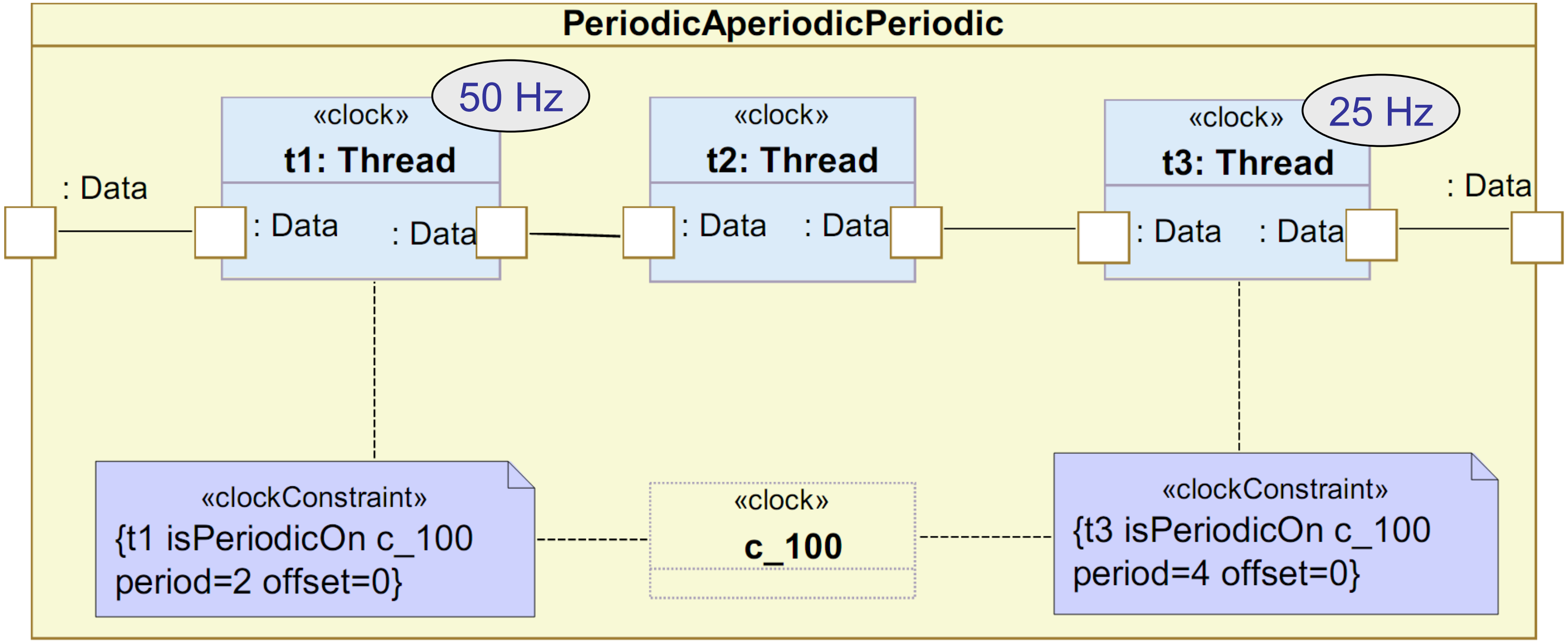


δ = 4

Visual representation ?

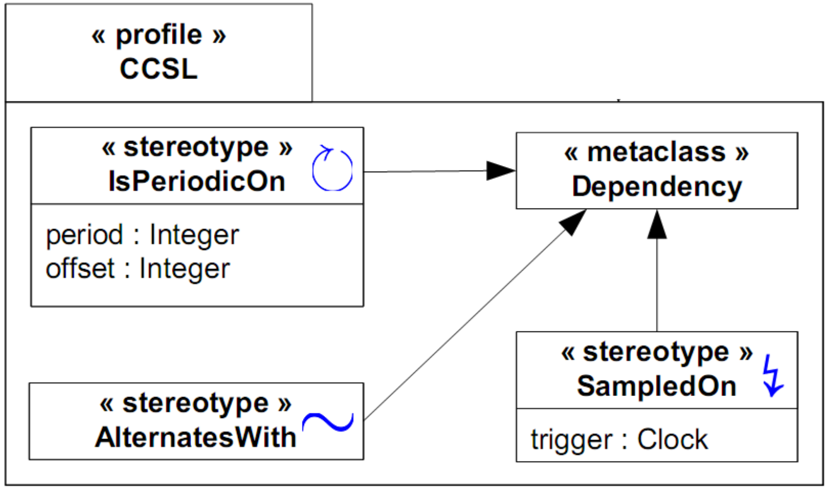
Illustration





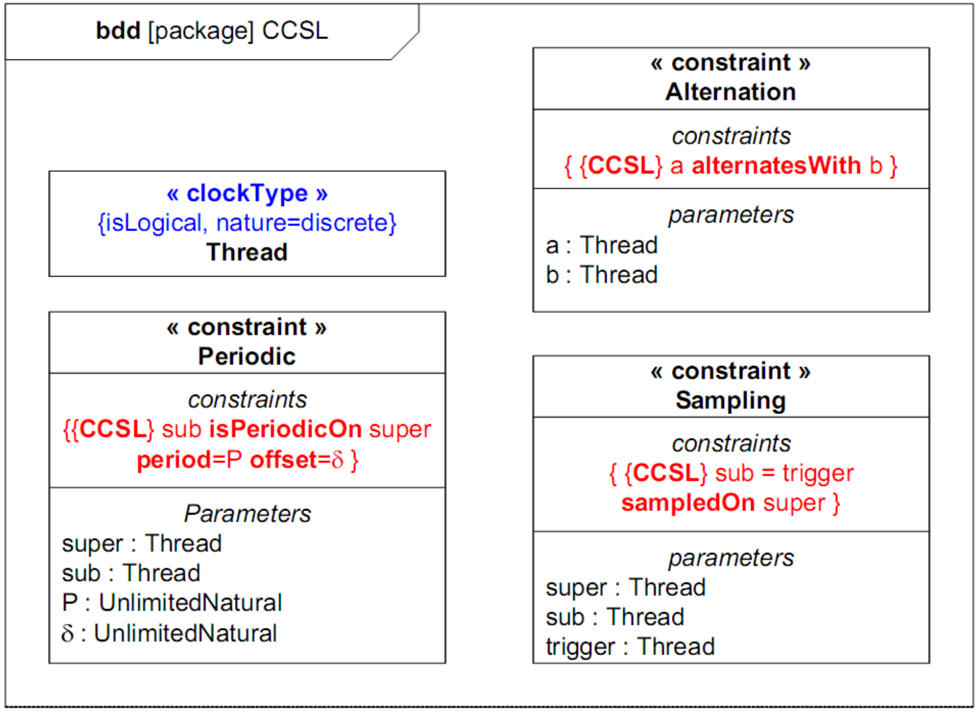
c_100 = IdealClk **discretizedBy** 0.01

UML Profile for CCSL



Requires to build another Profile on top of MARTE

SysML parametrics



bdd [package] CCSL

« clockType »
{isLogical, nature=discrete}
Thread

« constraint »
Periodic

constraints
{{**CCSL**} sub **isPeriodicOn** super
period=P **offset**=δ }

Parameters
super : Thread
sub : Thread
P : UnlimitedNatural
δ : UnlimitedNatural

« constraint »
Alternation

constraints
{ {**CCSL**} a **alternatesWith** b }

parameters
a : Thread
b : Thread

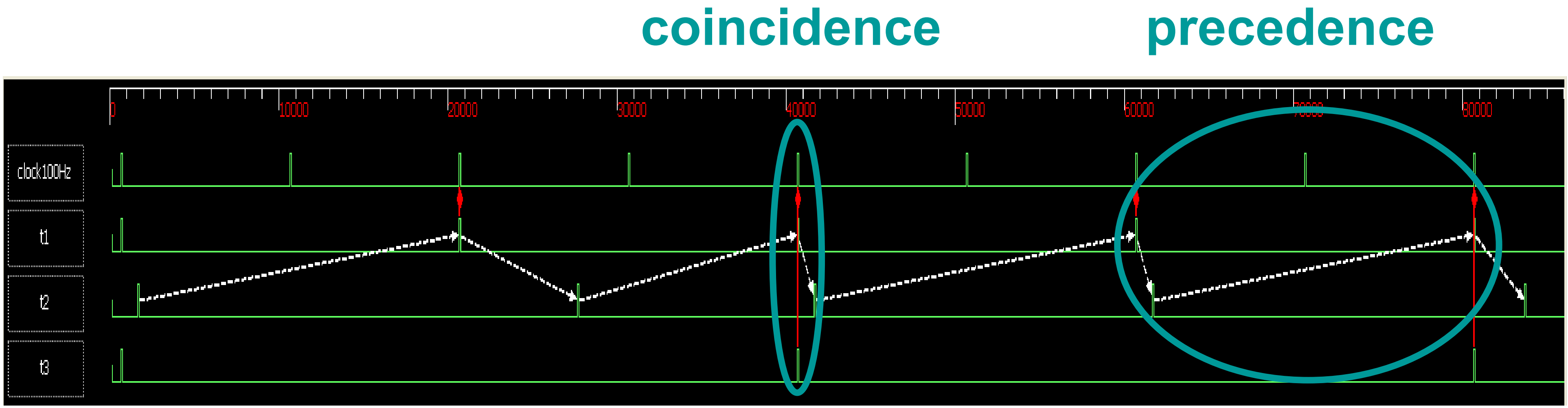
« constraint »
Sampling

constraints
{ {**CCSL**} sub = trigger
sampledOn super }

parameters
super : Thread
sub : Thread
trigger : Thread

Requires to define a SysML library for CCSL

CCSL simulator



- Produce VCD files (Verilog – IEEE Standard 1364 – 2005)
- TimeSquare (Eclipse plug-in available at : http://www-sop.inria.fr/aoste/)
- Constraints are evaluated step by step
  - Each step = set of boolean equations (SOS rewriting rules)
  - Builds the set of acceptable solutions : AllSAT
  - Simulation policy to choose one : random, min, max (asap), …

Conclusion

- Four categories of CCSL constraints
  - Synchronous
  - Asynchronous
  - Mixed
  - Quantitative

- Three alternative graphical representations
  - Stereotyped constraints
  - UML Profile for CCSL
  - Library of SysML constraint blocks

Questions

- UML & formal methods
  - Formal semantics should be part of UML
  - Can it appear in OMG specification ?
- The formal semantics should modify the graphical representation
  - Are profiles appropriated ?
  - Are meta-model tools more suitable ?